Compte rendu TP4

Global

On sépare en fichiers le main.c qui contient l'exécution propre à l'exercice et les fonctions qui pourraient être utilisées dans d'autres exercices et considérées comme une librairie (fonction.c).

Le fichier fonction.h permet de déclarer les fonctions en mettant les entetes, mais aussi de créer de nouveau types (notamment des structures pour l'exercice 2).

Exercice 1

Pour les différentes fonctions utilisées, il faut penser aux cas spécifiques à gérer.

Factorielle(x)

- 0 ! = 1
- X ne peut pas être négatif

Combinaison(n,p)

- N>=p
- N et P ne peuvent pas être négatifs

Permutation(*a,*b)

- Les pointeurs ne peuvent pas être nulls

Dans ces situations, il faut donc effecteur une vérification des arguments, une rectification parfois comme pour la combinaison afin d'avoir n>=p. Il faut également s'assurer que la conception des algorithmes permet la bonne gestion des conventions comme 0 !=1 (ou n^0=1).

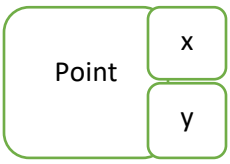
Des problèmes peuvent être rencontrés avec l'intervalle de valeurs possibles suivant les types. C'est pourquoi ici, le type long int a été choisi. Afin de proposer un intervalle plus large qui sera exploité principalement avec la fonction factorielle qui évolue en +infini de manière exponentielle.

On décide de mettre en fonction la demande de saisie d'une variable ainsi que l'affichage de ses quelques données exploitées (pour x et y) car l'opération a été demandé 2 fois, et elle pourra ainsi être demandée plus de fois en adaptant rapidement l'algorithme dans le main(). Chaque variable demandée est acceptée lorsqu'elle est >=0.

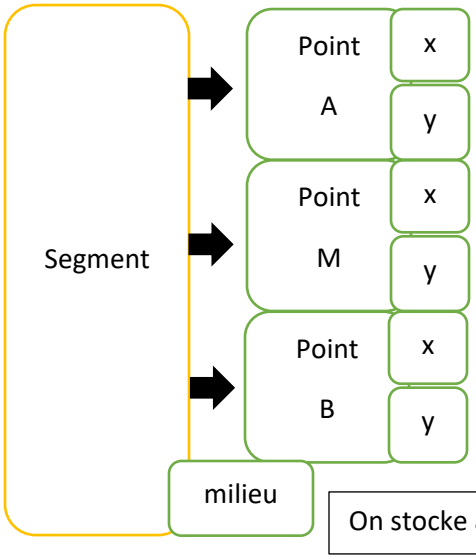
Exercice 2

Ici, on fait le choix d'utiliser des structures afin d'avoir des données faciles à exploiter avec une arborescence mais aussi une bonne lisibilité.

Nous créons ainsi la structure point.



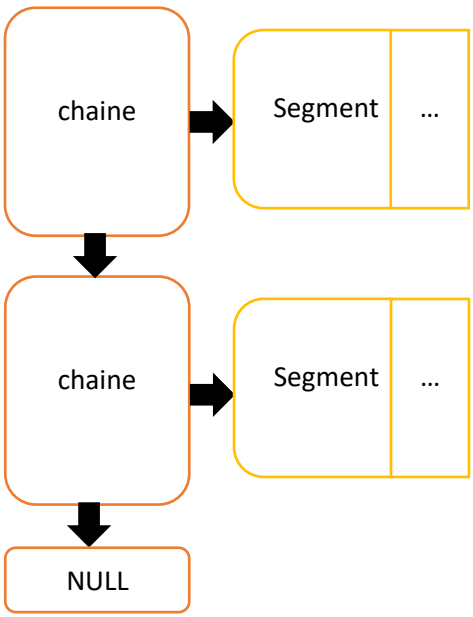
La structure segment



Ici, on fait le choix de stocker le point milieu du segment

On stocke aussi l'information sur le positionnement du milieu par rapport aux axes

Et une structure chaine qui nous servira pour l'exploitation des fonctions



Etant donné que l'on utilise des structures et que l'on ne sait pas combien de segments vont être entrés par l'utilisateur, on utilise de l'allocation dynamique de mémoire. Et afin de conserver l'accès à tous ces segments, on utilise une liste chainée.

On a donc 3 fonctions creerPoint, creerSegment, creerChaine qui nous permettent d'allouer de l'espace mémoire et d'initialiser les valeurs, on renvoie alors un pointeur sur l'objet en question.

Les fonctions demanderValeur, demanderPoint, demanderSegment permettent d'interroger l'utilisateur sur les valeurs qu'il veut et ainsi créer les objets avec une initialisation personnalisée.

On remarquera l'arborescence des demandes. La demande d'un segment va demander la demande de 2 points qui vont demander chacun 2 valeurs (x et y) pour les coordonnées.

C'est pourquoi ces fonctions ont des arguments permettant un affichage plus clair. (le numéro du segment, le nom du point, la coordonnée demandée).

Lors de la création d'un segment, on va également calculer le milieu en faisant appel à une autre fonction qui va prendre en arguments les coordonnées de A et B ainsi que des pointeurs sur les coordonnées du point M milieu que l'on veut initialiser.

La fonction calculerMilieu modifie la valeur de xM et yM par pointeur. Elle retourne une valeur 2,1 ou 0 en fonction de la localisation du point milieu sur les axes du repère. Cette valeur de retour sera stockée dans l'objet segment.

Pour l'exploitation des fonctions dans le main(), on s'assure de bien gérer la liste chainée (sauvegarde de la tête de liste, liaison de chaque chaine). Chaque chaine permet ainsi de sauvegarder un segment.

Lorsque l'utilisateur désire arrêter la saisie des segments, on procède à l'affichage.

Pour cela on parcourt la liste chainée en commençant par la tête et on s'arrête dès que l'on a une chaine NULL. L'affichage s'effectue simplement en montrant les coordonnées de chaque point du segment [A—M—B]. On fait le choix de garder 2 décimales.

On compte en même temps le nombre de milieux confondus à l'origine, le nombre de milieux étant sur un des 2 axes et ceux sur aucuns axes pour ensuite l'afficher.

Au fur et à mesure du parcours de la chaine, on en profite pour libérer l'espace mémoire alloué pour chaque point, segment et chaine.

Jeu de données permettant différents tests

A(0 ;0) B(0 ;0) // Sur l'origine ; A(4 ;7) B(8 ;-2) // Sur aucuns axes

A(-3 ;0) B(3 ;1) // Sur Ox ; A(4 ;2) B(8 ;-2) // Sur Oy